# PHP-2

## *1 Formulieren GET en POST*

datasource

## ? Leerdoelen

- Je weet wat `GET` en `POST` zijn.
- Je kunt gegevens doorsturen van een formulier naar een PHP-bestand.
- Je begrijpt het verschil tussen `GET` en `POST` qua werking en veiligheid.

Weet je nog wat formulieren zijn in PHP?

*Een formulier is een groep invoervelden in HTML. Deze velden wordne ingevuld en de waardes worden verstuurd naar een (meestal) andere pagina.*

## ? Uitleg

Deze regel in een HTML form is belangrijk

```
<form action="mijn-actie.php" method="get">
```

De `action` geeft aan **waar** het formulier de gegvens heen stuurt en de `method` geeft aan **hoe** deze gegevesn worden verstuurd.

Bij een formulier kies je of je `GET` of `POST` gebruikt om gegevens naar de server te sturen:

| Methode | Kenmerk | Voorbeeld |
|---|---|---|
| `GET` | Gegevens worden zichtbaar in de URL (minder veilig) | `mijn-actie.php?naam=Ali` |
| `POST` | Gegevens worden onzichtbaar verstuurd (veiliger) | De gegevens zitten in het HTTP-verzoek, niet in de URL |

## ?Voorbeeld

In dit voorbeeld wordt het formulier naar 'zichzelf' vertuurd.

https://www.youtube.com/watch?v=6AzAYU8AOhQ

https://www.youtube.com/embed/6AzAYU8AOhQ

# ?? Opdracht 1 – formulier.html (GET)

1. Maak een bestand `formulier.html` en zet daar deze code in:

```
<!DOCTYPE html>
<html>
<body>

  <h2>Wat is je naam?</h2>

  <form action="begroeting.php" method="get">
    <label for="naam">Naam:</label><br>
    <input type="text" id="naam" name="naam" required><br><br>
    <input type="submit" value="Verstuur">
  </form>

</body>
</html>
```

Let op de `method="get"` en het attribuut `required`.

# ?? Opdracht 2 – begroeting.php (GET)

1. Maak het bestand `begroeting.php` met de volgende PHP-code:

```
<?php
if (isset($_GET["naam"])) {
    $naam = $_GET["naam"];
    echo "<h1>Hallo $naam!</h1>";
} else {
    echo "<p>Geen naam ingevuld.</p>";
```

```
}
?>
```

**Uitleg:** We controleren met `isset()` of de waarde bestaat.

Vul het formulier in en kijk wat er gebeurt. Wat zie je op URL als je de begroeting ziet? Pas de tekst na het = teken op de url aan en reload je pagina, wat gebeurt er?

# ?? Opdracht 3 – POST-versie maken

1. Pas in `formulier.html` het formulier aan naar:

```
<form action="begroeting2.php" method="post">
```

2. Maak een nieuw bestand `begroeting2.php` met deze code:

```
<?php
if (isset($_POST["naam"])) {
    $naam = $_POST["naam"];
    echo "<h1>Welkom, $naam (via POST)</h1>";
} else {
    echo "<p>Geen naam ontvangen.</p>";
}
?>
```

# ? Reflectie

- Wat is het verschil tussen `POST` en `GET`?
- Wat gebeurt er precies in het PHP-bestand als je op “Verstuur” klikt?
- Wanneer zou je liever `POST` gebruiken dan `GET`? Geef een voorbeeld.

# ? Inleveren

- Lever de post-versie in van het formulier (.html).
- Beantwoord de drie reflectievragen en lever dit in als .txt of .pdf-bestand.

# *2 Include en Require*

## ? Leerdoelen

- Je weet wat `include` en `require` doen in PHP.
- Je kunt een header of footer inladen met `include`.
- Je herkent het verschil tussen `include` en `require`.

## ? Uitleg

Vaak gebruik je op meerdere pagina's dezelfde stukjes HTML, zoals een menu of een footer. Je kunt dat opslaan in een apart bestand en invoegen met `include` of `require`.

- `include "bestand.php"`: probeert het bestand in te laden. Als dat niet lukt, gaat de pagina wel gewoon verder.
- `require "bestand.php"`: probeert het bestand in te laden. Als dat niet lukt, stopt de pagina met een fout.

### Voorbeeld:

Stel, je maakt een bestand `header.php` met daarin een simpel menu:

```
<!-- header.php -->
<header>
  <h1>Mijn Website</h1>
  <nav>
    <a href="index.php">Home</a> |
    <a href="info.php">Info</a>
  </nav>
</header>
```

En dan gebruik je `include` in een andere pagina:

```
<?php include "header.php"; ?>

<h2>Welkom op de homepagina</h2>
<p>Dit is de inhoud van index.php</p>
```

# ?Voorbeeld

https://www.youtube.com/watch?v=aU0Tqxgtid8

https://www.youtube.com/embed/aU0Tqxgtid8

# ?? Opdracht – Header en footer maken

1. Maak een bestand `header.php` met een kop en menu zoals hierboven.
2. Maak een bestand `footer.php` met daarin bijvoorbeeld:
   `<footer>© 2025 Mijn Website</footer>`
3. Maak een bestand `index.php` met bovenin een `include("header.php")` en onderin een `include("footer.php")`.

# ? Reflectie

- Waarom is het handig om met `include` te werken?
- Wanneer zou je liever `require` gebruiken?
- Wat zou er gebeuren als je honderd pagina's hebt zonder `include` te gebruiken? In welk geval zou dat onhandig zijn?

# ? Inleveren

- Beantwoord de drie vragen n eigen woorden uit de reflectie en lever die in (.txt of .pdf bestand).

# *3 Arrays en Loops*

# ? Leerdoelen

- Je weet wat een array is in PHP.
- Je kunt een array maken met meerdere waarden.
- Je kunt een `foreach`-loop gebruiken om alle items te tonen.

# ? Uitleg

Een **array** is een soort lijstje waarin je meerdere dingen kunt bewaren, zoals hobby's of namen.

Een (gewone) variabele is plaats in get computergehuegen waarin je **één waarde** kan opslaan.

Een array is een variabele die **meer dan één waarde** kan bevatten.



**Variabele**

Een doos met één vak waar je één waarde in kan opslaan.

**$mijnNaam = "Max";**



**Array**

Een doos met meer vakjes waar je meerdere waarden in kan opslaan.

**$klasNamen = ["Evan", "Toby", "Enas", "Mohamed"];**

Je gebruikt een `foreach`-loop om elk item uit de array één voor één te gebruiken.

## Voorbeeld:

```
<?php
$hobbies = ["voetbal", "lezen", "gamen"];

foreach ($hobbies as $hobby) {
  echo "<p>Mijn hobby is: $hobby</p>";
}
?>
```

**Resultaat:**
Mijn hobby is: voetbal
Mijn hobby is: lezen
Mijn hobby is: gamen

# ?Voorbeeld

**Vanaf 0:30 -tot asan 06:30**

https://www.youtube.com/watch?v=D7mqV-p1kEc

https://www.youtube.com/embed/D7mqV-p1kEc

# ?? Opdracht – Eigen array

Maak een bestand `lijst.php` en vul dit met een array van 5 dingen die jij leuk vindt (hobby's, favoriete eten, games...).

- Gebruik een `foreach`-loop om ze allemaal op het scherm te tonen.
- Geef elk item weer in een eigen paragraaf (<p>).

# ? Uitleg

Er is ook een ander soort loop; een for-loop.

Gebruik een `for`-loop in plaats van `foreach`:

```
<?php
$games = ["Minecraft", "FIFA", "Fortnite", "Roblox"];
for ($i = 0; $i < count($games); $i++) {
  echo "<p>" . $i . " Favoriete game: " . $games[$i] . "</p>";
}
?>
```

De waarde van $games[0] is dus "Minecraft", $games[1] = "FIFA", etc, etc.

> Het eerste element in een array heeft een index 0!

# ?? Opdracht - deel 2

Plaats in de PHP-code (lijst.php) een eigen voorbeeld waarbij je een `for-loop` gebruikt.

# ? Reflectie

- Wat is het voordeel van een array ten opzichte van losse variabelen?
- Wat is het verschil tussen `for` en `foreach`?
- Wanneer zou je liever een `for`-loop gebruiken?

## ? Inleveren

- Lever een screenshot in van jouw `lijst.php` met minimaal 5 items in een array, getoond in de browser.
- Laat zien dat je zowel `foreach` als `for` gebruikt hebt.

# *4 Loops met indexed arrays*

## ? Leerdoelen

- Je weet wat een indexed array is in PHP.
- Je kunt een array maken en vullen met meerdere waarden.
- Je kunt met een `foreach`-loop door een array lopen.
- Je kunt gegevens uit een array weergeven met `echo`.

## ? Uitleg

Een **indexed array** is een lijst waarbij elk element een nummer (index) heeft, beginnend bij 0. Bijvoorbeeld:

```
<?php
$producten = ["Chips", "Cola", "Tandpasta", "Zeep"];
?>
```

Met een `foreach`-loop kun je door deze array heen lopen en elk item laten zien:

```
<?php
foreach ($producten as $product) {
  echo "<p>$product</p>";
}
?>
```

## ?? Opdracht – Winkelmandje weergeven

**Situatie:** Je bouwt een simpele webwinkel. Je hebt een lijst met producten die iemand in zijn winkelmandje heeft geplaatst. Jij zorgt dat de producten netjes op het scherm getoond worden met

behulp van een array en een loop.

1. Maak een nieuw bestand aan met de naam `winkelmandje.php`.
2. Maak een indexed array met minimaal 5 producten uit een supermarkt (bijv. "Brood", "Melk", ...).
3. Gebruik een `foreach`-loop om de lijst als HTML-lijst (`<ul>`) weer te geven.
4. Tel hoeveel producten er in het winkelmandje zitten met `count()` en toon dat onderaan.

## Voorbeelduitvoer:

- Brood
- Melk
- Appels

**Totaal:** 3 producten

# ? Reflectie

- Waarom is het handig om een lijst in een array op te slaan?
- Wat gebeurt er als je een nieuw product toevoegt aan de array? Moet je je loop aanpassen?
- Hoe zou je deze code hergebruiken als iemand anders een andere boodschappenlijst heeft?

# ? Inleveren

- Lever je bestand `winkelmandje.php` in.
- Lever je reflectie in als .txt of .pdf.

# *5 (Associative) Arrays en Loops*

# ? Leerdoelen

- Je begrijpt het verschil tussen indexed en associatieve arrays.

- Je kunt beide soorten arrays aanmaken in PHP.
- Je kunt met een `foreach`-loop informatie uit arrays tonen.

# ? Uitleg

## Indexed array

Een indexed array gebruikt nummers als '**key**'. Dit is handig voor eenvoudige lijsten:

```
<?php
$namen = ["Fatima", "Noah", "Aziz"];
echo $namen[0];
echo "<br>";
echo $namen[1];
echo "<br>";
echo $namen[2];
echo "<br>";
?>
```

## Associatieve array

Een associatieve array gebruikt zelfgekozen '**keys**'. Dit is handig voor gegevens met betekenis:

```
<?php
$student = [
  "voornaam" => "Fatima",
  "achternaam" => "Bakker",
  "email" => "fatima.bakker@example.com"
];
echo $student['voornaam'];
echo "<br>";
echo $student['achternaam'];
echo "<br>";
echo $student['email'];
echo "<br>";
?>
```

# ?Voorbeeld

Vanaf 0:30 -tot asan 06:30 had je al bekeken eerder, **nu ga je kijken vanaf 08:55 - 12:00**

https://www.youtube.com/watch?v=D7mqV-p1kEc

https://www.youtube.com/embed/D7mqV-p1kEc

# ??Opdracht – Studentenlijst maken

Test beide voorbeelden.

1. onder aan de code druk je nog een keer alle waarden af, maar dan door geburik te maken va een **loop**.
2. Voeg dan op regel 2 in de eerste code je eigen naam toe en test je code nog een keer. Wat gebeurt er?
3. Voeg dan tussen achternaam en email voor de student de woonplaats toe. Geef deze een waarde. Test je code nog een keer. Wat gebeurt er?

# ? Reflectie

- Wat is het verschil tussen indexed en associatieve arrays?
- Wat gebeurt er als je een 'key' verkeerd typt?
- In de opdracht heb je kunnen zien waarom je beter een loop kan gebruiken om een array af te drukken. Leg in eigen woorden uit waarom.

# ? Inleveren

- Lever je reflectie in als .txt of .pdf.

# *6 Functies in PHP*

# ? Leerdoelen

- Je weet wat een functie is in PHP.

- Je kunt zelf een functie schrijven en aanroepen.
- Je kunt gegevens meegeven aan een functie (parameters).

# ? Uitleg

Een functie is een blokje code dat je een naam geeft en later opnieuw kunt gebruiken. Zo hoef je niet telkens dezelfde code opnieuw te schrijven.

Je kunt informatie aan een functie meegeven (dat noem je een **parameter**) en je kunt iets teruggeven (dat noem je een **return**).

## Voorbeeld:

```
<?php
function begroet($naam) {
  echo "Hallo, $naam!<br>";
}

begroet("Fatima");
begroet("Ali");
?>
```

# ?Voorbeeld

https://www.youtube.com/watch?v=SBFO5bIDYHQ&t=185s

https://www.youtube.com/embed/SBFO5bIDYHQ?t=185s

# ?? Opdracht 1 – Maak je eigen begroetfunctie

1. Maak een nieuw bestand `functie.php`
2. Maak daarin een functie `begroet` die één parameter accepteert: `$naam`
3. Laat de functie een boodschap tonen zoals “Hoi, [naam], welkom terug!”
4. Roep de functie minstens 3 keer aan met verschillende namen.

# ?? Opdracht 2 – Maak een rekentool met functie

1. Maak een functie `kortingBerekenen` die twee getallen als parameter gebruikt: `$bedrag` en `$percentage`
2. Laat de functie het bedrag na korting `returnen`
3. Laat het resultaat op het scherm zien met `echo`

## Voorbeeldcode:

```
<?php
function kortingBerekenen($bedrag, $korting) {
  $nieuwBedrag = $bedrag - ($bedrag * ($korting / 100));
  return $nieuwBedrag;
}

echo "<p>Je betaalt nu: €" . kortingBerekenen(100, 25) . "</p>";
?>
```

# ? Reflectie

- Waarom is het handig om functies te gebruiken?
- Wat gebeurt er als je geen parameter meegeeft aan een functie die er wel één verwacht?
- Waar zou je functies nog meer voor kunnen gebruiken?

# ? Inleveren

- `functie.php` met ten minste 1 begroetfunctie én 1 rekentool.
  Je moet ten minste 2 functies gebruiken met parameters, waarvan 1 ook een `return` gebruikt.

# *7 Datum en Tijd in PHP*

# ? Leerdoelen

- Je weet hoe je de huidige datum en tijd kunt tonen met PHP.
- Je kunt berekenen hoeveel dagen of jaren geleden iets was.
- Je kunt de leeftijd berekenen op basis van een geboortedatum.

# ? Uitleg

PHP heeft functies om met datum en tijd te werken. De belangrijkste is `date()`, waarmee je de huidige tijd kunt weergeven in verschillende formaten.

- `date("Y")` → jaar (bijv. 2025)
- `date("d-m-Y")` → dag-maand-jaar (bijv. 04-06-2025)
- `date("H:i")` → tijd in uren:minuten (bijv. 14:36)

## Voorbeeld:

```
<?php
echo "Vandaag is het: " . date("d-m-Y") . "<br>";
echo "Het is nu: " . date("H:i") . " uur.";
?>
```

# ?? Opdracht 1 – Toon de datum en tijd

1. Maak een bestand `datum.php`
2. Laat daarin zien:
   - Welke dag het vandaag is
   - De datum in formaat `dd-mm-yyyy`
   - De tijd in uren en minuten

# ?? Opdracht 2 – Leeftijd berekenen

Bereken iemands leeftijd op basis van geboortedatum:

```
<?php
$geboortedatum = "2007-03-15";
$geboortedag = new DateTime($geboortedatum);
$vandaag = new DateTime();
```

```
$verschil = $vandaag->diff($geboortedag);
echo "Je bent " . $verschil->y . " jaar oud.";
?>
```

**Uitleg:** PHP kan automatisch het aantal jaren berekenen tussen twee datums met `diff()`.

## ? Reflectie

- Wat is het verschil tussen `date()` en `new DateTime()`?
- Waarom moet je bij het berekenen van leeftijd een `diff()` gebruiken?
- Wat zou er gebeuren als je geboortedatum in een verkeerd formaat schrijft?

## ? Inleveren

- `datum.php` met de juiste datum, tijd en berekende leeftijd.
  Je gebruikt minstens één functie met datum en één met tijd, en `diff()` voor de leeftijd.

# *8 Sessies en Inloggen met PHP*

## ? Leerdoelen

- Je weet wat een sessie is in PHP.
- Je kunt een eenvoudige inlogpagina maken.
- Je kunt een gebruiker herkennen op een andere pagina.

## ? Uitleg

Een **sessie** in PHP is een manier om informatie te onthouden zolang de gebruiker je website bezoekt. Bijvoorbeeld of iemand is ingelogd of wat zijn/haar gebruikersnaam is.

Je start een sessie met:

```
<?php
session_start();
?>
```

# ?Voorbeeld

https://www.youtube.com/watch?v=JXKQmIpjRTk

https://www.youtube.com/embed/JXKQmIpjRTk

# ?? Opdracht 1 – inloggen.html

Maak een bestand `inloggen.html` met een formulier waarin je gebruikersnaam invoert:

```
<!DOCTYPE html>
<html>
<body>

  <h2>Inloggen</h2>

  <form action="login.php" method="post">
    <label for="naam">Naam:</label><br>
    <input type="text" name="naam" id="naam" required><br><br>
    <input type="submit" value="Inloggen">
  </form>

</body>
</html>
```

# ?? Opdracht 2 – login.php

Maak een bestand `login.php` dat de naam opslaat in een sessie en doorstuurt:

```
<?php
session_start();
$_SESSION["gebruiker"] = $_POST["naam"];
header("Location: welkom.php");
exit;
?>
```

# ?? Opdracht 3 – welkom.php

Maak een bestand `welkom.php` dat de gebruiker begroet:

```
<?php
session_start();
if (isset($_SESSION["gebruiker"])) {
  echo "<h1>Welkom, " . $_SESSION["gebruiker"] . "!</h1>";
  echo '<p><a href="uitloggen.php">Uitloggen</a></p>';
} else {
  echo "<p>Je bent niet ingelogd.</p>";
}
?>
```

# ?? Opdracht 4 – uitloggen.php

```
<?php
session_start();
session_destroy();
header("Location: inloggen.html");
exit;
?>
```

# ? Reflectie

- Waarom moet je altijd `session_start()` gebruiken bovenaan?
- Wat gebeurt er als je probeert `welkom.php` te openen zonder in te loggen?
- Wat zou je kunnen uitbreiden, bijvoorbeeld met wachtwoordcontrole?

# ? Inleveren

- Screenshots van `inloggen.html`,
- screenshot van `welkom.php` met jouw naam.
- Leg uit (in .txt) waarom je geen screenshot kan maken van de web pagina uitloggen.php.

Je moet gebruik maken van sessies en de naam van de gebruiker correct kunnen tonen op de pagina’s.

# *9 Inloggen met wachtwoordcontrole*

## ? Leerdoelen

- Je maakt een formulier met gebruikersnaam én wachtwoord.
- Je controleert de invoer in PHP.
- Je begrijpt waarom `$_GET` niet veilig is voor wachtwoorden.
- Je leert werken met een associatieve array.

## ? Uitleg

Een loginformulier stuurt gebruikersnaam en wachtwoord naar PHP. In deze les gebruiken we eerst `$_GET` om te laten zien waarom dat niet veilig is – je ziet het wachtwoord in de URL.

## ?? Opdracht 1 – login.html

Maak een bestand `login.html`:

```
<!DOCTYPE html>
<html>
<body>

  <h2>Loginformulier</h2>

  <form action="controle.php" method="get">
    <label>Gebruikersnaam:</label><br>
    <input type="text" name="gebruiker" required><br><br>

    <label>Wachtwoord:</label><br>
    <input type="password" name="wachtwoord" required><br><br>

    <input type="submit" value="Log in">
  </form>

</body>
</html>
```

# ?? Opdracht 2 – controle.php

Maak een bestand `controle.php` waarin je controleert of de gebruiker mag inloggen:

```
<?php
$gebruiker = $_GET["gebruiker"];
$wachtwoord = $_GET["wachtwoord"];

if ($gebruiker == "admin" && $wachtwoord == "geheim123") {
  echo "<h2>Welkom, $gebruiker!</h2>";
} else {
  echo "<p>Foutieve inloggegevens. Probeer opnieuw.</p>";
}
?>
```

## Let op:

Als je dit formulier verstuurt, zie je het wachtwoord in de URL. Dat is niet veilig!

# ?? Opdracht 3 – Verbeter met POST

- Pas het formulier aan zodat het `method="post"` gebruikt
- Pas `controle.php` aan zodat het `$_POST` gebruikt
- Test: zie je het wachtwoord nog in de URL?

# ? Uitleg – Associatieve array

Tot nu toe heb je gewerkt met lijsten zoals:

```
$hobby’s = ["voetbal", "gamen", "lezen"];
```

Dit is een **indexed array**: de computer onthoudt zelf de volgorde (index 0, 1, 2).

Een **associatieve array** heeft zelfgekozen namen als index (zogenaamde "keys"):

```
$gebruikers = [
  "admin" => "geheim123",
  "student" => "welkom01"
];
```

Je kunt dan bijvoorbeeld zeggen:

```
echo $gebruikers["admin"]; // toont: geheim123
```

Heel handig voor wachtwoorden of gebruikerslijsten!

# ?? Extra opdracht – Meerdere gebruikers

Breid `controle.php` uit met een associatieve array van toegestane gebruikers en wachtwoorden:

```
<?php
$gebruikers = [
  "admin" => "geheim123",
  "student" => "welkom01",
  "docent" => "phprules"
];

$gebruiker = $_POST["gebruiker"];
$wachtwoord = $_POST["wachtwoord"];

if (isset($gebruikers[$gebruiker]) && $gebruikers[$gebruiker] == $wachtwoord) {
  echo "<h2>Welkom, $gebruiker!</h2>";
} else {
  echo "<p>Inloggen mislukt.</p>";
}
?>
```

**Voeg zelf nog twee gebruikers toe: één voor jouw zelf (dus je eigen voornaam) en één voor een klasgenoot.**

# ? Reflectie

- Wat is het verschil tussen een indexed array en een associatieve array?
- Waarom is het veiliger om `$_POST` te gebruiken voor wachtwoorden?
- Wat zou je doen om inloggen met een wachtwoord nog veiliger te maken?

# ? Inleveren

- controle.php

- Een .txt. of .pdf bestand met de antwoorden op de drie reflectievragen.

---

Revision #24
Created 2025-06-04 19:31:20 UTC by Max
Updated 2026-07-08 18:14:24 UTC by Max